Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims

Claims 1 through 12. (Cancelled)

13. (Previously Presented) In a computer network comprising two or more arbitrarily defined data structures, each of the data structures comprising one or more leaf data elements, a description of each data structure being known, a system for transferring data from a first in-memory data component corresponding to a first data structure to a second in-memory data component corresponding to a second data structure, the system comprising:

a) a mapping tool configured to:

allow a user to graphically define a relation and association between leaf data elements of a first data structure description and leaf data elements of a second data structure description; and

generate one or more mapping descriptions of the relation and association between the leaf data elements of the first data structure description and the leaf data elements of the second data structure description as defined by the user; and

b) a high-performance run-time engine configured to:

dynamically generate a first in-memory data component containing actual data associated with the leaf data elements of the first data structure description using a key-based look-up molding technique, the first in-memory data component comprising at least one lookup table;

traverse the one or more mapping descriptions and accessing the at least one lookup table of the first in-memory data component to get actual data stored in the first in-memory data component using a key-based look-up technique;

dynamically generate a second in-memory data component configured to store actual data associated with the leaf data elements of the second data structure description using the key-based look-up molding technique, the second in-memory data component comprising at least one lookup table; and

transfer data between the first in-memory data component and the second in-memory data component according to the one or more mapping descriptions.

14. (Previously Presented) The system as recited in claim 13, wherein the mapping tool is further configured to store the one or more mapping descriptions in a machine readable format.

15. (Previously Presented) The system as recited in claim 14, wherein the machine readable format is XML.

16. (Previously Presented) The system as recited in claim 13, wherein the one or more mapping descriptions is expressed using a unique path identifier with an absolute and relative path addressing scheme.

17. (Previously Presented) The system as recited in claim 13, further comprising a cache configured to store the one or more mapping descriptions and configured to respond to multiple requests for accessing the one or more mapping descriptions.

18. (Previously Presented) The system as recited in claim 13, wherein the mapping tool is further configured to allow a user to graphically define a one-to-one relation and association between the leaf data elements of the first data structure description and the leaf data elements of the second data structure description.

19. (Previously Presented) The system as recited in claim 13, wherein the mapping tool is further configured to allow a user to graphically define a one-to-many relation and association between the leaf data elements of the first data structure description and the leaf data elements of the second data structure description.

20. (Previously Presented) In a distributed computer system comprising one or more data structures, a method for storing and retrieving actual data of a data structure in order to use the actual data of the data structure to perform data transfer functions, the method comprising:

identifying a hierarchy of one or more data containers in a first data structure description, wherein a data container can be defined as either a singular data container or a plural data container, wherein at least one of the data elements in the one or more data containers is a leaf data element;

traversing the hierarchy of one or more data containers in the first data structure description to determine a unique key for each leaf data element of the first data structure description, comprising at least one of:

determining whether all of the one or more data containers in the first data structure description are singular data containers, wherein, for each leaf data element, a key is generated containing a concatenation of all names of the data containers in a hierarchical path to the leaf data element, each data container name separated by a character that is not allowed as part of the data container name, concatenated with a name of the leaf data element, and storing the key in a lookup table of a single in-memory data component; or

determining whether one or more data containers in the first data structure description is a plural data container, wherein upon identifying a plural data container, a component list is instantiated in a lookup table having a key that is generated containing a concatenation of names of all the data containers traversed either from a root node or from a previous plural data container to a hierarchical path to the identified plural data container, the component list comprising a plurality of data components.

21. (Previously Presented) The method as recited in claim 20, further comprising, upon identifying a plural data container, further comprising, for each leaf data element, generating a key containing a concatenation of names of all the data containers traversed from the previous plural data container to the hierarchical path of the leaf data element, each data container name separated by a character that is not allowed as part of the data container name,

concatenated with a name of the leaf data element, and storing the key in one of the plurality of data components.

22. (Previously Presented) The method as recited in claim 20, further comprising inserting a marker object as the value of the generated key.

23. (Previously Presented) The method as recited in claim 22, wherein the marker object is one of a string, a static object of a class, or a unique integer value.

24. (Previously Presented) The method as recited in claim 20, further comprising

storing in cache one or more lookup tables containing the first data structure description;

receiving a request for generating another lookup table for the first data structure description; and

returning a copy of the cached one or more lookup tables for the first data structure description.

25. (Previously Presented) The method as recited in claim 20, further comprising modifying actual data of at least one leaf data element of the first data structure description by including in a request a key generated for the leaf data element and sending the request to a lookup table of the leaf data element key to modify a data field corresponding to the leaf data element.

26. (Previously Presented) The method as recited in claim 20, further comprising

using a mapping tool to define a relation and association between leaf data elements of the first data structure description and leaf data elements of a second data structure description; and

automating transfer of the actual data associated with the leaf data elements of the first data structure description stored in the first in-memory data component to the second in-memory data component based on a mapping description of the relation and association between leaf data elements of the first data structure description and leaf data

elements of the second data structure description, further comprising accessing one or more lookup tables of the first in-memory data component.

27. (Previously Presented) The method as recited in claim 26, further comprising storing in memory the mapping description of the relation and association between leaf data elements of the first data structure description and leaf data elements of the second data structure description.

28. (Previously Presented) The method as recited in claim 26, wherein the mapping description of the relation and association between leaf data elements of the first data structure description and leaf data elements of the second data structure description includes identification of built-in-functions, further comprising transferring the actual data associated with the leaf data elements of the first data structure description stored in the first in-memory data component to the second in-memory data component using the built-in functions.

29. (Previously Presented) The method as recited in claim 26, wherein the mapping description of the relation and association between leaf data elements of the first data structure description and leaf data elements of the second data structure description includes leaf data elements from the first data structure description being flattened.

30. (Previously Presented) The method as recited in claim 20, further comprising enforcing a well-defined set of rules to restrict users as to data types that can be used to define the data elements of the first data structure description, the enforceable restrictions being one or more of whether the data type is singular or plural, a default value for the data type, whether the data type indicates that a corresponding data value is required at runtime, a data range for the data type, allowed data values for the data type, a data format for the data type, or other enforceable restrictions for each data type.

31. (Previously Presented) A computer program product for use in a system having a processor, the computer program product comprising a computer usable medium having computer readable program code stored thereon, the computer readable program code comprising computer executable instructions that, when executed by a processor, cause the computer program product to perform the following:

map a relation and association between leaf data elements of a first data structure description and leaf data elements of a second data structure description;

set data values of any leaf data element of the first data structure description in a first in-memory data component using a key generated from a key-based look-up molding technique;

get the data values of any leaf data element of the first data structure from the first in-memory data component using the key generated from the key-based look-up molding technique; and

automate transfer of the data values from the first in-memory data component to a second in-memory data component.

32. (Previously Presented) The method as recited in claim 31, wherein mapping a relation and association between leaf data elements of a first data structure description and leaf data elements of a second data structure description further comprises storing in memory the first data structure description in one or more lookup tables.

33. (Previously Presented) The method as recited in claim 31, wherein the mapping description is expressed in a machine readable format using a unique path identifier with an absolute and relative path addressing scheme.

34. (Previously Presented) The method as recited in claim 31, wherein setting data values of any leaf data element of the first data structure description in a first in-memory data component using a key-based look-up molding technique further comprises storing a key corresponding to each leaf data element in one or more lookup tables.

35. (Previously Presented) The method as recited in claim 31, further comprising enforcing integrity of the data values while and after the data values are set.

36. (Previously Presented) The method as recited in claim 31, further comprising enforcing software interface integrity, data type validation and enforcing data type restrictions at runtime.

37. (Previously Presented) The method as recited in claim 31, wherein the first data structure and the second data structure are both software services, wherein mapping a relation and association between leaf data elements of a first data structure description and leaf data elements of a second data structure description comprises mapping the outputs of the first software service with the inputs of the second software service.

38. (Cancelled)